

# 《The road to AGI》

YKY 甄景贤

July 15, 2020

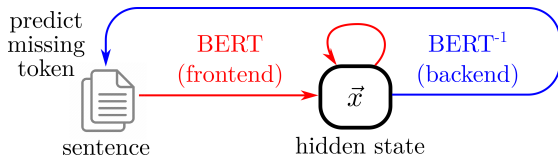
# Table of contents

- 1 BERT 的革命性意义:「闭环路训练」
- 2 BERT 的内部结构
- 3 BERT 赋予逻辑结构
- 4 「集」结构 带来麻烦
- 5 Attention 是什么?
- 6 Attention 给逻辑 AI 的启发
- 7 Attention... is not what we want
- 8 「神经」知识表示
- 9 神经 特征簇 (feature clusters)
- 10 高阶 特徵
- 11 关于“model-based reasoning”的质疑
- 12 神经  $\leftrightarrow$  逻辑 correspondence
- 13 再谈一次 逻辑结构
- 15 Type theory and the Curry-Howard isomorphism
- 16 Topos theory and fibrations

多谢 支持 😊

# BERT 的革命性意义：「闭环路训练」

- BERT 利用平常的文本 induce 出知识，而这 representation 具有 通用性 (universality)：



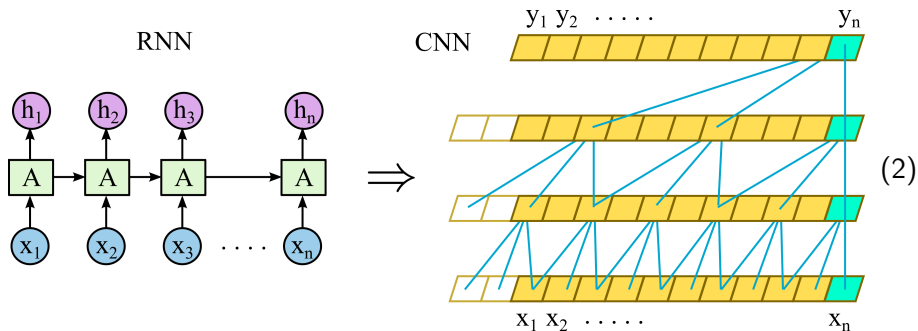
换句话说：隐状态的 representation 压缩了句子的意思，而它可以应用在别的场景下

- This implies that human-level AI can be *induced* from existing corpora, 而不需重复 像人类婴儿成长的学习阶段
- Such corpora can include items such as images, movies with dialogues / subtitles
- 这种训练方法是较早的另一篇论文提出，它并不属于 BERT 的内部结构

# BERT 的内部结构

其实，BERT 也是混合了很多技巧 发展而成的：

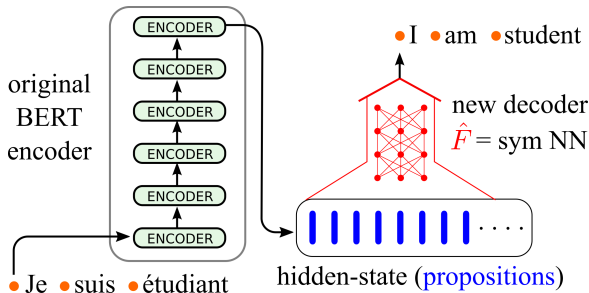
- BERT 基本上是一个 seq-to-seq 的运算过程
- Seq-to-seq 问题最初是用 RNN 解决的
- 但 RNN 速度较慢，有人提出用 CNN 取代：



- CNN 加上 attention mechanism 变成 Transformer
- 我的目标是重复这个思路，但引入 symmetric NN 的结构

# BERT 赋予逻辑结构

- 逻辑结构 可以用 对称化的 的 BERT 模型 处理：



(3)

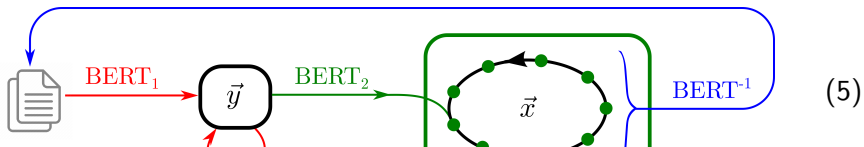
- 这时，输出对 **|** 的交换不变性 is automatically satisfied by the architecture of the decoder
- 而如果假设 BERT's encoder is a universal seq-to-seq mapping, 似乎满足了对称性的要求（还要考虑 注意力机制 有什么影响，这一点我仍未有时间想）
- 当然最好还是用实验证实 🤔

# 「集」结构 带来麻烦

- 词语 组成 句子，类比於 逻辑中，概念 组成 逻辑命题
- 抽象地说，逻辑语言 可以看成是一种有 2 个运算的 代数结构，可以看成是 加法  $\wedge$  和 乘法  $\cdot$ ，其中 乘法 是不可交换的，但加法 可交换
- 例如 两个命题：

$$\text{我} \cdot \text{爱} \cdot \text{妳} \wedge \text{妳} \cdot \text{爱} \cdot \text{我} \quad (4)$$

- Word2Vec 也是革命性的；由 Word2Vec 演变成 Sentence2Vec 则比较容易，基本上只是 向量的 延长 (concatenation)；逻辑命题 类似於 sentence
- 假设 全体逻辑命题的空间是  $\mathbb{P}$ ，则 命题集合 的空间是  $2^{\mathbb{P}}$ ，非常庞大
- 如果限制 状态  $\vec{x}$  = working memory 只有 10 个命题， $\vec{x}$  的空间是  $\mathbb{P}^{10} / \sim$  其中  $\sim$  是对称群  $\mathfrak{S}_{10}$  的等价关系。换句话说  $2^{\mathbb{P}} \cong \coprod_{n=0}^{\infty} \mathbb{P}^n / \mathfrak{S}_n$
- $\mathbb{P}^n / \mathfrak{S}_n$  虽然是  $\mathbb{P}^n$  的商空间，但  $\mathfrak{S}_n$ -不变性 很难用神经网络实现
- 现时 比较可行的办法，是将 状态  $\vec{x}$  实现成 一个时间上的「轮盘」，每个  $\bullet$  表示一个命题：



# Attention 是什么？

- 注意力 最初起源於 Seq2seq, 后来 BERT 引入 self-attention
- 在 Seq2seq 中, 编码器 (encoder) 由下式给出, 它将输入的词语  $x_i$  转化成一连串的 隐状态  $h_i$ :

$$h_t = \text{RNN}_{\text{encode}}(x_t, h_{t-1}) \quad (6)$$

- 这些  $h_i$  可以综合成单一个 隐状态  $c = q(h_1, \dots, h_n)$ .
- 这个  $c$  被「寄予厚望」, 它浓缩了整个句子的意义
- 解码器 的结构类似, 它的隐状态是  $s_t$ , 输出  $y_t$ :

$$s_t = \text{RNN}_{\text{decode}}(y_t, s_{t-1}, c_t) \quad (7)$$

- 注意最后的  $c_t$  依赖时间, 它是隐状态  $h_j$  的 加权平均:

$$c_i = \sum_j \alpha_{ij} h_j \quad (8)$$

- 其中  $\alpha_{ij}$  量度 输入 / 输出 的隐状态之间的 相似度, 取其最大值:

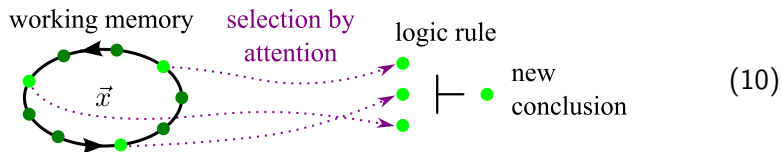
$$\alpha_{ij} = \text{softmax}\{\langle s_i, h_j \rangle\} \quad (9)$$

换句话说,  $\alpha_{ij}$  选择 最接近  $h_j$  的  $s_i$

# Attention 给逻辑 AI 的启发

我这样理解 attention:

- 例如, 翻译时, 输入 / 输出句子中「动词」的位置可以是不同的
- 当 解码器需要一个「动词」时, 它的隐状态  $s_t$  含有「动词」的意思
- Attention 机制 找出最接近「动词」的 编码器的隐状态 (可以  $\geq 1$  个)  $\sum h_j$ , 交给 解码器, 这是一种 information retrieval
- 例如, 将  $M$  件东西 映射 到  $N$  件东西, 可以有  $N^M$  个 mappings, 这是非常庞大的空间。但如果这些物件有 类别, 而 同类只映射到同类, 则可以用 attention 简化 mappings
- 所以 attention 是一种 inductive bias, 它大大地缩小 mapping 空间
- 在逻辑的场景下, 需要的 mapping 是  $f: \text{命题集合} \rightarrow \text{命题}$





## Attention... is not what we want

- 逻辑 attention 和 传统 attention 要求略有不同，这是关键的一步
- 不是「同类映射到同类」，而是要在庞大的 logic rules 空间中找到适用 (applicable) 的 rules
- 隐状态  $s_t$  代表 “search state”，注意力 的目的是 选择  $s_t$  所需要的那些命题，交给 解码器
- 注意：逻辑 attention 从  $M$  个命题中 选择  $N$  个命题， $M > N$ . 这是 inductive bias. 而 Symmetric NN 的做法，只是要求  $M$  个命题 的 置换不变性，所以它浪费了资源在很多 “don't care” 的命题上
- 换句话说，selection 所带来的 bias 如果足够强，似乎不需要 symmetric. 很巧合地，再次应验了 “attention is all you need” 这句话

# 「神经」知识表示

为什么要研究 神经知识表示？

- 从经典 logic-based AI 的传统，一直在使用「符号」的知识表示法
- 符号逻辑 很容易转换成 抽象代数 / 范畴论 形式（它们是同一个大家庭的「近亲」）
- 然而 或许存在 截然不同 的知识表示法？但我们很难想像它 长什么样子
- 人脑的「神经」知识表示，可以作为参考，然后再研究它和逻辑表示之间的 correspondence

神经知识表示 的特点：

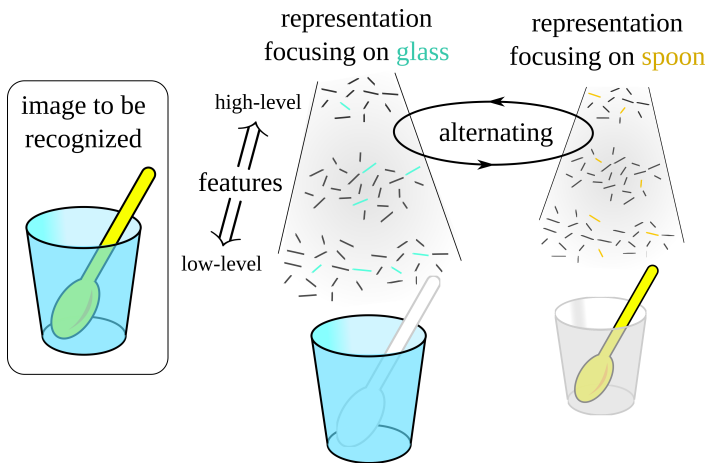
- distributive（分布性）
- model-based (vs rule-based)
- *in situ*（固定性）— 例如辨认「猫」的时候，大脑中 相应的神经元被 激活，但这些 神经元 不能移动，所以「猫」的表示 也不可移动

问题是：如果要辨认「白猫追黑猫」，「猫」的表示是固定的，则这两个「猫」表示 如何共存於神经网络中？

答案很可能是：两个「猫」交替地 出现在 时间上

# 神经 特征簇 (feature clusters)

例如，以「匙羹在杯中」作例子：

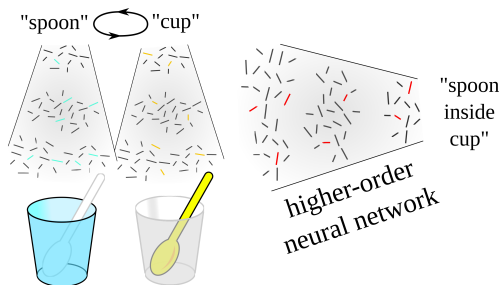


(11)

每个 复杂物体 由一个 **feature cluster** 辨认。多个「特征簇」在时间上交替出现，可以看成是一种 composition，例如  $A \cdot B$  或  $A \circ B$ 。

# 高阶 特徵

- 一串 特征簇 的时间序列，例如  $A \cdot B$ ，可以被 更高阶 的神经网络 用作输入。高阶辨认 的结果是一些关系 (relations)，例如「匙羹在杯内」



(12)

- 这似乎是一个 特征空间  $\times$  时间 的映射  $f: X \times T \rightarrow Y$
- 关于这部分其实我仍未肯定，或许有其他方法

# 關於 “model-based reasoning” 的質疑

- 很多人认为大脑的思考方式是 先在脑中构造 models，然后再从 models 中「读出」一些结论
- 例如给定一个描述：「已婚妇人出轨，用刀刺死丈夫」

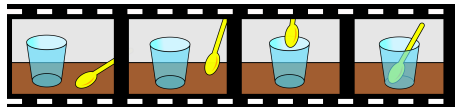


(13)

- 如果假设「妻子有长头发」、「丈夫死时穿著西装」，这些都是 臆想 出来的细节，是不正确的
- 那么这 model 可以有哪些细节？答案是：任何细节都不可以有，除非是 逻辑上蕴含的
- 例如我们可以假设妻子 probably 有一双手臂，但也有例外的情况是独臂的，这是一种 逻辑推导
- 所以，其实所谓 “model-based reasoning” 并没有那么神奇，也并不一定正确，它的细节必需被 逻辑 约束
- 而 model 本身也可以用一些 抽象的逻辑命题 构成，这也是合理的；反而，一个有很多感官细节的 model 并不合理

# 神经 $\leftrightarrow$ 逻辑 correspondence

- 我们的目标是了解 神经表示 和 逻辑表示 之间的关系，这关系或许可以用 范畴论描述？
- 定义 复杂情境 (complex scenario) 是 感知材料 (sensory data) 的一个片段，如：



(14)

又或者一个故事，例如「John 爱 Mary 但 Mary 不爱他」

- 一个复杂情境 可以用若干个 特征簇 描述
- Equivalently, 复杂情境 可以用 逻辑 表示，就是一大堆 逻辑命题 的 conjunction，这些命题 钜细无遗 地描述该情境

## 再谈一次 逻辑结构

- 以前曾经说过，机器视觉 的成功，有赖於 将 视觉的几何结构 impose 在 深度神经网络上
- 这 深度神经网络 原本是“free”的，但加了限制之后，权重空间 变小了（例如维数降低），所以学习加速了
- 所谓 symmetry 的意义，简单例子：「如果知道左边等於右边，那就只需计算一次」
- 换句话说，数学家喜欢对称性，是因为它经常可以简化计算
- 同理，我们想将 逻辑结构 的对称性 impose 到神经网络
- 实际上，可能只需要逻辑上的交换律，就可以达到 强人工智能，正如 机器视觉的成功，在於引入了 CNN 的 convolution 结构，后者只是 视觉不变性 的其中一个最显著的 invariant
- 现代逻辑理论 非常漂亮，我花了十多年时间才弄懂，我希望将这套 逻辑-学习 理论简单讲解一下，也算功德圆满了

- 在经典时代，逻辑的代数形式可以用 Boolean algebra 表述，然而这方法只适用於命题逻辑
- Boolean algebra 是中學生熟悉的，类似 Venn diagram 的结构
- 这种结构和拓撲学的 open sets 结构一样，所以命题逻辑也可以看成是一种 topology
- 然而 predicate logic 的结构更复杂，直到最近才有比较完善的表述
- 现代逻辑结构和 type theory 有深刻的关系，此即 Curry-Howard isomorphism
- 现代逻辑也涉及 topos theory，那是一种由 algebraic geometry 引入的结构



# Type theory and the Curry-Howard isomorphism

- 大家都知道 Lisp 语言没有 type, 它是一种 untyped  $\lambda$ -calculus
- 在 Lisp 之上引入 type system, 衍生成 ML, Caml, OCaml, Haskell 等一系列语言
- 每一个 program 属于某个 type, 例如 `length()` 函数, 输入一个字串, 输出它的长度;  $\text{length} : \text{String} \rightarrow \text{Integer}$
- 有些逻辑学家 察觉到 类型论 的  $\tau_1 \rightarrow \tau_2$  和逻辑中  $P \rightarrow Q$  是一模一样的
- 这个关系的发现者至少包括:  
Brouwer-Heyting-Kolmogorov-Curry-de Bruijn-Howard
- 这关系的深刻之处, 在于把 符号逻辑上的 proofs 和 程式语言的 programs 划上等号, 前者是 符号 / 静态的, 后者是 程序 / 动态的
- 每个 proof 就是一个 program, 它输入一些 arguments, 输出 关于那些 arguments 的证明
- 例如: 「所有人都会死」是一个 program, 它输入「苏格拉底」, 输出「苏格拉底会死」
- 这个对应也许可以应用到深度学习: 神经网络 也是一种 函数 / mapping, 它将 逻辑前提 map 到结论

# Topos theory and fibrations

- Predicate logic (谓词逻辑) 和 命题逻辑 之间的差异在於 fibration 结构
- Fibration 通常用  $\begin{smallmatrix} \mathbb{E} \\ \downarrow \pi \\ \mathbb{B} \end{smallmatrix}$  表示,  $\mathbb{B}$  = base space,  $\mathbb{E}$  = entire space,  $\pi$  = projection
- Base space 是 type 的空间, entire space 是 predicate 的空间
- 由於 Curry-Howard 对应, type = propositions, 在 base 空间上只有 命题逻辑
- 例如  $\mathbb{B}$  空间的一个 type 是 Human,  $\mathbb{E}$  空间的一个谓词是 Mortal
- 於是有以下这个 type inference rule:

$$i : \text{Human} \vdash \text{Mortal}(i) : \text{Prop} \quad (15)$$

意思是说, 如果  $i$  屬於 Human 类型, 则  $\text{Mortal}(i)$  屬於 Prop 类型



- map proofs to proofs, is this OK in the propositional case?
- what is a proof anyway?
- What is the proof of  $H(a)$ ? Is it  $a \in H$  or  $a : H$ ?
- $H$  could be a type but it is not a single proposition.  $H(a)$  would be a type.
- Then what is the proof of  $H(a)$ ? Perhaps the tuple  $(H, a)$ ?

多谢收看 😊